Amendments to the specification,

Marked version of the replacement paragraph(s)/section(s), pursuant to 37 CFR 1.121(b)(1)(ii):

Please replace paragraph [9] with the following rewritten paragraph:

Owing to their digital nature, computers essentially only understand "machine code," i.e., the low-level, minute instructions for performing specific tasks -- the sequence of ones and zeros that are interpreted as specific instructions by the computer's microprocessor. Since machine language or machine code is the only language computers actually understand, all other programming languages represent ways of structuring human language so that humans can get computers to perform specific tasks. While it is possible for humans to compose meaningful programs in machine code, practically all software development today employs one or more of the available programming languages. The most widely used programming languages are the "high-level" languages, such C++, Pascal, or more recently Java® and C#. These languages allow data structures and algorithms to be expressed in a style of writing that is easily read and understood by fellow programmers.

Please replace paragraph [11] with the following rewritten paragraph:

The ultimate output of the compiler is a compiled module such as a compiled C++ "object module," which includes instructions for execution ultimately by a target processor, or a compiled Java® class, which includes bytecodes for execution ultimately by a Java® virtual machine. A Java® compiler generates platform-neutral "bytecodes" -- an architecturally neutral, intermediate format designed for deploying application code efficiently to multiple platforms.

Please replace paragraph [12] with the following rewritten paragraph:

Integrated development environments, such as Borland's JBuilder® ~~(registered trademark)~~, Delphi (trademark) and C# Builder (trademark), are the preferred application development environments for quickly creating production applications. Such environments are characterized by an integrated development environment (IDE) providing a form painter, a property getter/setter manager ("inspector"), a project

manager, a tool palette (with objects which the user can drag and drop on forms), an editor, a debugger, and a compiler. In general operation, the user "paints" objects on one or more forms, using the form painter. Attributes and properties of the objects on the forms can be modified using the property manager or inspector. In conjunction with this operation, the user attaches or associates program code with particular objects on the screen (e.g., button object). Typically, code is generated by the IDE in response to user actions in the form painter and the user then manipulates the generated code using the editor. Changes made by the user to code in the editor are reflected in the form painter, and vice versa. After the program code has been developed, the compiler is used to generate binary code (e.g., Java® bytecode) for execution on a machine (e.g., a Java® virtual machine).

Please replace paragraph [14] with the following rewritten paragraph:

"Refactoring" is a practice of making structured changes to software applications or systems which add the desired flexibility, but keep the functionality of the system the same. Refactoring involves taking small individual steps that are well defined and that can be applied in succession to yield more significant changes. For example, a developer may wish to perform a "rename refactoring" to change the name of a particular module (e.g., a class name in a Java® program). In order to make this change, the user must locate the definition of this class (i.e., the source code for the class) as well as all uses of the class in other portions of the system. In the case of a class name in a Java® program, the class name is typically used not only for defining a variable, but also for constructing instances (or objects) of that class and accessing static members of the class (i.e., class variables). Another example of refactoring may involve moving a specified class to a new package (referred to as "move refactoring").

Please replace paragraph [31] with the following rewritten paragraph:

Fig. 3A is a block diagram of a Java® development system suitable for implementing the present invention.

Please replace paragraph [40] with the following rewritten paragraph:

Bytecode: A virtual machine executes virtual machine low-level code instructions called bytecodes. Both the Sun Microsystems Java® virtual machine and the Microsoft .NET virtual machine provide a compiler to transform the respective source program (i.e., a Java® program or a C# program, respectively) into virtual machine bytecodes.

Please replace paragraph [41] with the following rewritten paragraph:

Compiler: A compiler is a program which translates source code into binary code to be executed by a computer. The compiler derives its name from the way it works, looking at the entire piece of source code and collecting and reorganizing the instructions. Thus, a compiler differs from an interpreter which analyzes and executes each line of code in succession, without looking at the entire program. A "Java compiler" translates source code written in the Java® programming language into bytecode for the Java® virtual machine. For general background on the construction and operation of compilers, see e.g., Fischer et al., "Crafting a Compiler with C", Benjamin/Cummings Publishing Company, Inc., 1991, the disclosure of which is hereby incorporated by reference for purposes of illustrating the state of the art.

Please replace paragraph [42] with the following rewritten paragraph:

Java: Java® is a general purpose programming language developed by Sun Microsystems. Java® is an object-oriented language similar to C++, but simplified to eliminate language features that cause common programming errors. Java® source code files (files with a .java extension) are compiled into a format called bytecode (files with a .class extension), which can then be executed by a Java® interpreter. Compiled Java® code can run on most computers because Java® interpreters and runtime environments, known as Java® virtual machines (VMs), exist for most operating systems, including UNIX, the Macintosh OS, and Windows. Bytecode can also be converted directly into machine language instructions by a just-in-time (JIT) compiler. Further description of the Java® Language environment can be found in the technical, trade, and patent literature; see e.g., Gosling, J. et al., "The Java Language Environment: A White Paper," Sun Microsystems Computer Company, October 1995, the disclosure of which is hereby incorporated by reference. For additional information on the Java® programming

language (e.g., version 2), see e.g., "Java 2 SDK, Standard Edition Documentation, version 1.4.2," from Sun Microsystems, the disclosure of which is hereby incorporated by reference. A copy of this documentation is available via the Internet (e.g., currently at java.sun.com/j2se/1.4.2/docs/index.html).

Please replace paragraph [56] with the following rewritten paragraph:

Java® development environment

Please replace paragraph [57] with the following rewritten paragraph:

Java® is a simple, object-oriented language which supports multi-thread processing and garbage collection. Although the language is based on C++, a superset of C, it is much simpler. More importantly, Java® programs are "compiled" into a binary format that can be executed on many different platforms without recompilation. A typical Java® system comprises the following set of interrelated technologies: a language specification; a compiler for the Java® language that produces bytecodes from an abstract, stack-oriented machine; a virtual machine (VM) program that interprets the bytecodes at runtime; a set of class libraries; a runtime environment that includes bytecode verification, multi-threading, and garbage collection; supporting development tools, such as a bytecode disassembler; and a browser (e.g., Sun's "Hot Java" browser).

Please replace paragraph [58] with the following rewritten paragraph:

Fig. 3A is a high-level block diagram illustrating a Java® development system 300 suitable for implementing the present invention. As shown, the Java® development system 300 includes a client 310 which employs a virtual machine 320 for executing programs. In particular, the client 310 executes a "compiled" (i.e., bytecode or pseudo-compiled) Java® program 340, which has been created by compiling a Java® source code program or script 305 with a Java® compiler 330. Here, the Java® source code program 305 is an application program written in the Java® programming language; the pseudo-compiled program 340, on the other hand, comprises the bytecode emitted by the compiler 330. The virtual machine 320 includes a runtime interpreter for interpreting the Java® bytecode program 340. During operation, the client 310 simply requests the

virtual machine 320 to execute a particular Java® compiled program.

Please replace paragraph [67] with the following rewritten paragraph:

Runtime support libraries 324 comprise functions (typically, written in C) which provide runtime support to the virtual machine, including memory management, synchronization, type checking, and interface invocation. At the client machine on which a Java® application is to be executed, runtime support libraries 324 are included as part of the virtual machine; the libraries are not included as part of the Java® application. The bytecode which is executed repeatedly calls into the runtime support libraries 324 for invoking various Java® runtime functions.

Please replace paragraph [68] with the following rewritten paragraph:

In the currently preferred embodiment, the Java® development system 300 may be provided by Borland® JBuilder® ~~(registered trademark)~~ 10.0, available from Borland Software Corporation of Scotts Valley, CA. Further description of the development system 300 may be found in "Building Applications with JBuilder (JBuilder 10)" (Part No. JXE0010WW21005bajb), also available from Borland Software Corporation, the disclosure of which is hereby incorporated by reference.

Please replace paragraph [69] with the following rewritten paragraph:

The above-described computer hardware and software are presented for purposes of illustrating the basic underlying desktop and server computer components that may be employed for implementing the present invention. For purposes of discussion, the following description will present examples in which it will be assumed that there exists at least one computer running applications developed using the Java® programming language. The present invention, however, is not limited to any particular environment or device configuration. In particular, use of the Java® programming language is not necessary to the invention, but is simply used to provide a framework for discussion. Instead, the present invention may be implemented in any type of system architecture or processing environment capable of supporting the methodologies of the present invention presented in detail below. The following description will focus on those features of the

development system 300 which are helpful for understanding the methodology of the present invention for asynchronous refactoring.